Validating Requirements for Fault-Tolerant Systems Using Model Checking

Francis Schneider, Jack Callahan, Steve Easterbrook

The objectives of this work were to develop new techniques for validating fault-tolerance requirements in the early stages of software development, and then determining whether the requirements and high-level designs for fault-tolerant systems provide the required reliability before they are implemented. Another purpose was to explore the integrated application of model-checking technology as a verification and validation technique for software requirements.

A case study of a dually redundant spacecraft controller, in which a checkpoint and rollback scheme is used to provide fault tolerance during the execution of critical control sequences, was conducted. The software-requirements specification for the spacecraft established the required behavior for the checkpoint and rollback scheme. Their validity could not be determined through inspection. In other words, it was not possible to determine whether the behavior described in these requirements would provide the desired level of fault tolerance. Testing of the eventual implementation would not necessarily provide this validation either, a result of the difficulty of ensuring test-case coverage for all possible fault-occurrence scenarios. The approach taken was to derive a formal automata-based model from the specification, and to use a model-checker to explore its behavior. Various high-level safety properties were used to validate the generalized system model. Key system functional requirements were validated by using linear temporal logic to define the corresponding liveness properties, which are required to be satisfied when the system responds to faults. The model checker, Spin, identified traces in the model for which these properties were violated, using nondeterministic fault injection.

Validation of fault-tolerant architectures is a difficult problem, and exhaustive testing of the implemented systems is an unsatisfactory approach to its solution. Errors found after implementation are expensive to fix. If the fault-tolerant architecture is found to be deficient during system testing, then much of the development effort may have been wasted. Testing cannot guarantee coverage of all possible fault conditions, for the precise timing of fault occurrences can determine how they are handled. For these reasons, techniques that can be applied earlier in the life cycle are needed.

Model checking can be applied to abstract models of the proposed architecture early in the life cycle, and can explore model behavior in the presence of a wide variety of fault conditions. A model was abstracted from design notes for the dual-redundant system. The model was pruned to remove states that did not affect the properties to be tested, thereby reducing the size of the state space to one that is manageable by current model-checking tools. Five different fault categories were identified, and six separate requirements on the rollback scheme were validated. Each of the requirements involved the exhaustive examination of approximately 100,000 states in the model, and took about 30 seconds. The response and recovery in each case was to the injection of a fault of the appropriate category in all possible ways, based on the model.

Three of the six runs for the six requirements failed in the verification. Three anomalies were identified: two were errors in the requirements that might not occur in the implementation, and the third was a discrepancy in the detailed requirements that could allow for erroneous behavior of the implemented system. This analysis demonstrated that the approach is feasible, and that it is capable of detecting subtle errors that had escaped detection through other means.

Point of Contact: S. Easterbrook
(304) 367-8352
steve@research.ivv.nasa.gov